Edinburgh Napier UNIVERSITY

CASCADING STYLE SHEETS

Web Tech
SET08101

Simon Wells
s.wells@napier.ac.uk
http://www.simonwells.org

TL/DR

- HTML gives us structure (which is essential)
- CSS gives us presentation
  - Not just making things look pretty
  - But important to usability & accessibility
- Presentation from a technical perspective

AIMS

- At the end of this (sub-section) of the topic you will be able to:
  - Explain why structure & presentation are treated separately
  - Understand how HTML & CSS are related
  - Have a basic grasp of the syntax & semantics of CSS

As with HTML (& later Javascript) we can only scratch the surface of what each technology can do. This is a foundation. Exploring what it can do ***for you*** *is your responsibility*

Edinburgh Napier
UNIVERSITY

CSS

- Cascading Style Sheets (CSS)
- Simple, text-based, page appearance description language
- Early HTML mixed content and presentation
  - Every element needed font, colour, style, alignment, border, size, etc. explicitly described, often repeatedly so throughout HTML
  - Moving style declarations gives simpler HTML and more manageable design
- Needed a consistent & flexible way to control presentation
- Permit almost every HTML tag to be arbitrarily scaled, positioned, & decorated - overcomes limitations of underlying markup language

VERSIONS

- Currently version 3
    - CSS level 1drafted in 1996 (revisions until 2008)
    - CSS level 2 drafted in 1998 (revisions until 2011)
    - CSS level 3 drafted in 2005 (revisions continuing…)
- Most features of CSS2 & 3 supported by modern browsers
- Details can vary between implementations - mainly due to lack of finalised standard

- Moved from (pre CSS) HTML presentational attributes:

  <h1><font color="red">The Quick Brown Fox</font></h1>

- to style **parameters**:

  <h1 style="color: red;">The Quick Brown Fox</h1>

- to style **blocks**:

```
<!DOCTYPE html>
<html>
    <head>
        <style type="text/css">
            h1 {color: red;}
        </style>
    </head>
    <body>
        <h1>The Quick Brown Fox</h1>
    </body>
</html>
```

- to external **style sheets**:

  <link href="path/to/file.css" rel="stylesheet" type="text/css">

CONTENT & PRESENTATION

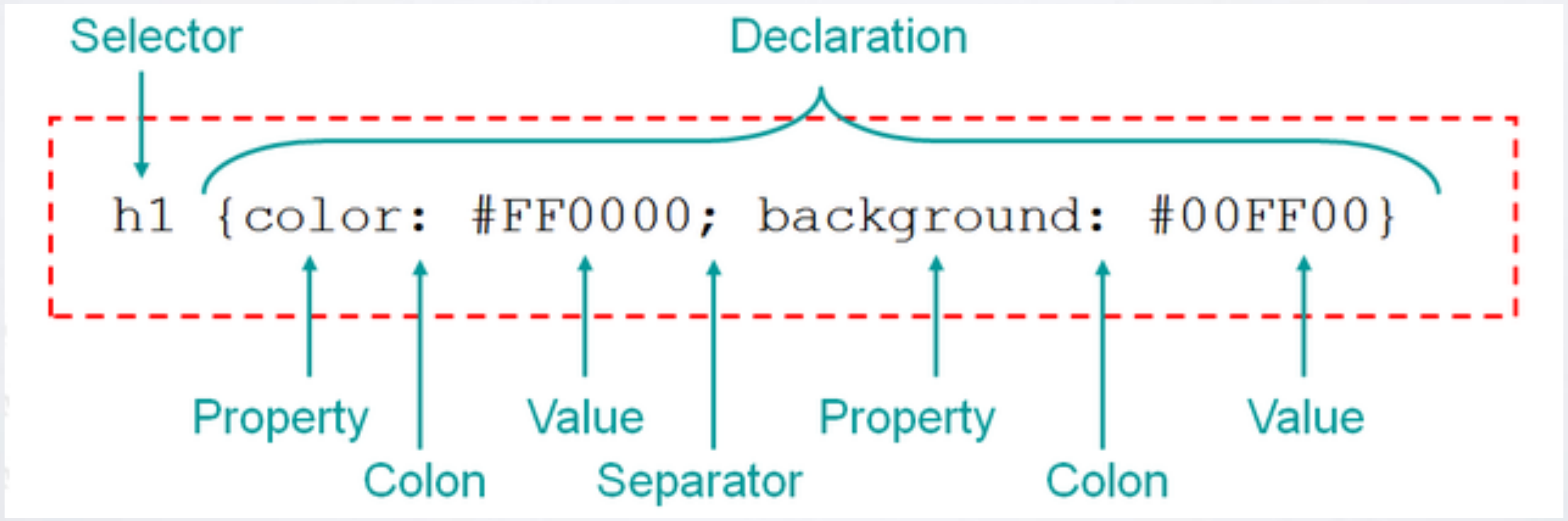
- Visual & design aspects separated from the core content & structure of the document
  - Think of human body:
    - Skeleton gives structure
    - Flesh gives appearance
- Not a rigid rule but more of a best practice:
  - Different members of a team can work on each aspect
  - Content can be presented in different ways
    - People with visual impairment can provide their own stylesheets to the browser to override the website developer’s decisions
    - Screen & Print versions of pages can be styled differently - tailor content to the medium of consumption
  - You *can* use inline presentation elements (or mix inline & separate) but can lead to future problems:
    - Separation leads to simplified change management

CHICKEN OR EGG?

- Design first or content first?
- Generally you can't design something well until you have some idea of what the design is working with - you need some content (or at least an idea of the structure of the content)
  - Content can drive the design process
    - One reason why we started with HTML and thinking about Data before getting to CSS
- Relate to other design ideas:
  - "Form follows function" - how something is structured stems primarily from the underlying engineering requirements
- We should not conform structure of content for design reasons if it leads to a compromised system, e.g. buggy, inefficient, unreliable, unusable
- But we can let structure inform the design process - many elegant solutions stem from the design exposing underlying structure

BASIC CSS SYNTAX

- Simple syntax
- Stylesheets consists of a list of rules
- Each rule (or rule set) consists of one or more selectors & a declaration block



SELECTORS

- Tells the browser which part of the markup to apply a style to, e.g. matching style to HTML tags and attributes
- Can apply to:
    - All elements of a specific type, e.g. <h1> or <img>
    - Elements specified by attribute but often
        - specified by *id* attribute, e.g. <p id="para-123"> #para-123
        - specified by *class* attribute, e.g. .photos selector applied to tags with <p *class="photos"*>. *.photos*
        - *NB.* pseudo-classes refer to special states of selected elements, e.g. hover, visited, active
            - Enables elements to be styled in relation to things outside the DOM, e.g. history of navigation
    - Elements depending upon their placement relative to others in the DOM
- Can be combined and joined in many ways to specify elements by location, element type, id, class (or any combination thereof)
- Order of selectors is important

DECLARATIONS

- Works with the selector - tells the browser the set of properties to apply to the elements selected by the selector
- A Declaration comprises a property, a colon (:), and a value, e.g.

color: red

- Properties defined in CSS standard & has a given set of values (keywords like "center" or values, e.g.
    - Colour can specified with keywords, e.g. red, or Hex values (#FF0000) or RGB values (rgb(255, 0, 0))
- Get used to consulting documentation:
    - Some CSS properties can affect any type of element. Others apply to particular groups of elements.
    - Each CSS property can take only a specific range of values
- Multiple declarations are separated by semi-colons (;)

color: red; text-align: center;

- A Declaration block is a list of declarations in braces

h1 {color: red; text-align: center;}

SOURCES

- CSS can be used with HTML in 3 primary ways:
    1. Attached to a specific tag using the *style* parameter
    2. Inlined **globally** using a <*style*> block
    3. Retrieved from an external URL using <*link* *rel*="*stylesheet*" *type*="*text/css*" *href*=“*theme.css*”> directive
- 2 & 3 require a *fully qualified stylesheet* consisting of any number of selectors

INHERITANCE

- A key CSS feature
- HTML parsed into DOM
- DOM is a tree which is hierarchical
- Nested descendants generally inherit text-related properties from parent elements that enclose them
- Efficient because don't have to declare same properties repeatedly

Given:

```
h1 { color: purple; }
```

But no declaration for the colour of the <em> element

```
<h1>
  This is to <em>illustrate</em> inheritance
</h1>
```

The <em> element would inherit the h1 property

USING THE STYLE PARAMETER

- "inline" with HTML Element attributes, e.g.
  - <p style="color:red">
- Advantage:
  - Put it right where you're using it
- Disadvantage:
  - Lots of repetition - have to specify everything
  - Mixture of content & presentation

Edinburgh Napier UNIVERSITY

USING A <STYLE> BLOCK

- Collect all styles declarations together
- Separates presentation from content
- Styles individual pages (but not sites)

```
<!DOCTYPE html>
<html>
    <head>
        <style type="text/css">
            h1 {color: red;}
        </style>
    </head>
    <body>
        <h1>The Quick Brown Fox</h1>
    </body>
</html>
```

Edinburgh Napier UNIVERSITY

EXTERNAL STYLESHEET

- Separation of content (in .html file) from presentation (in .css file)
- Can reuse the same .css file in multiple .html files
    - So can style an entire site (multiple HTML files) with all the presentation in one place
        - Easy to manage & update the design without touching the content
        - Can add new content & take advantage of ready made style
- NB. media attribute lets you specifiy different style sheets for different contexts, e.g. print, projection, aural, braille, tty, tv

index.html

```
<!DOCTYPE html>
<html>
    <head>
        <link href=“style.css" rel="stylesheet" type=“text/css" media=“screen”>
    </head>
    <body>
        <h1>The Quick Brown Fox</h1>
    </body>
</html>
```

style.css

```
body {background-color:black}
h1 {color:red}
p {color:blue}
```

COLOURS

- An easy place to start, visual impact.
- Control colour of element using the colour property
- 140 named colours:
    - lightcoral, rosybrown, indianred, red, firebrick, brown, darkred, maroon, mistyrose, salmon, tomato, darksalmon, coral, orangered, lightsalmon, sienna, seashell, chocolate, saddlebrown, sandybrown, peachpuff, peru, linen, bisque, darkorange, burlywood, antiquewhite, tan, navajowhite, blanchedalmond, papayawhip, moccasin, orange, wheat, oldlace, floralwhite, darkgoldenrod, goldenrod, cornsilk, gold, lemonchiffon, khaki, palegoldenrod, darkkhaki, ivory, lightyellow, beige, lightgoldenrodyellow, yellow, olive, olivedrab, yellowgreen, darkolivegreen, greenyellow, chartreuse, lawngreen, darkseagreen, honeydew, palegreen, lightgreen, lime, limegreen, forestgreen, green, darkgreen, seagreen, mediumseagreen, springgreen, mintcream, mediumspringgreen, mediumaquamarine, aquamarine, turquoise, lightseagreen, mediumturquoise, azure, lightcyan, paleturquoise, aqua, cyan, darkcyan, teal, darkslategray, darkturquoise, cadetblue, powderblue, lightblue, deepskyblue, skyblue, lightskyblue, steelblue, aliceblue, dodgerblue, lightslategray, slategray, lightsteelblue, cornflowerblue, royalblue, ghostwhite, lavender, blue, mediumblue, darkblue, midnightblue, navy, slateblue, darkslateblue, mediumslateblue, mediumpurple, blueviolet, indigo, darkorchid, darkviolet, mediumorchid, thistle, plum, violet, fuchsia, magenta, darkmagenta, purple, orchid, mediumvioletred, deeppink, hotpink, lavenderblush, palevioletred, crimson, pink, lightpink, white, snow, whitesmoke, gainsboro, lightgray, silver, darkgray, gray, dimgray, black,
- Each can be referred to by name, hex code, RGB or HSL code
- More colours available by code than are named (~ 16M+)

Edinburgh Napier UNIVERSITY

BACKGROUND

- background-color
- background-image
- background-repeat {repeat, repeat-x, repeat-y, no-repeat}
- background-position - specify two values for horizontal & vertical from {top, bottom, left, right, center}

FONTS PROPERTIES

- After colour, typography is an important consideration
  - font-family - which font to use
  - font-size - how big it should be
  - font-style - {normal, italic, oblique}
  - font-weight - {normal, bold, bolder, lighter, +*numeric values*}

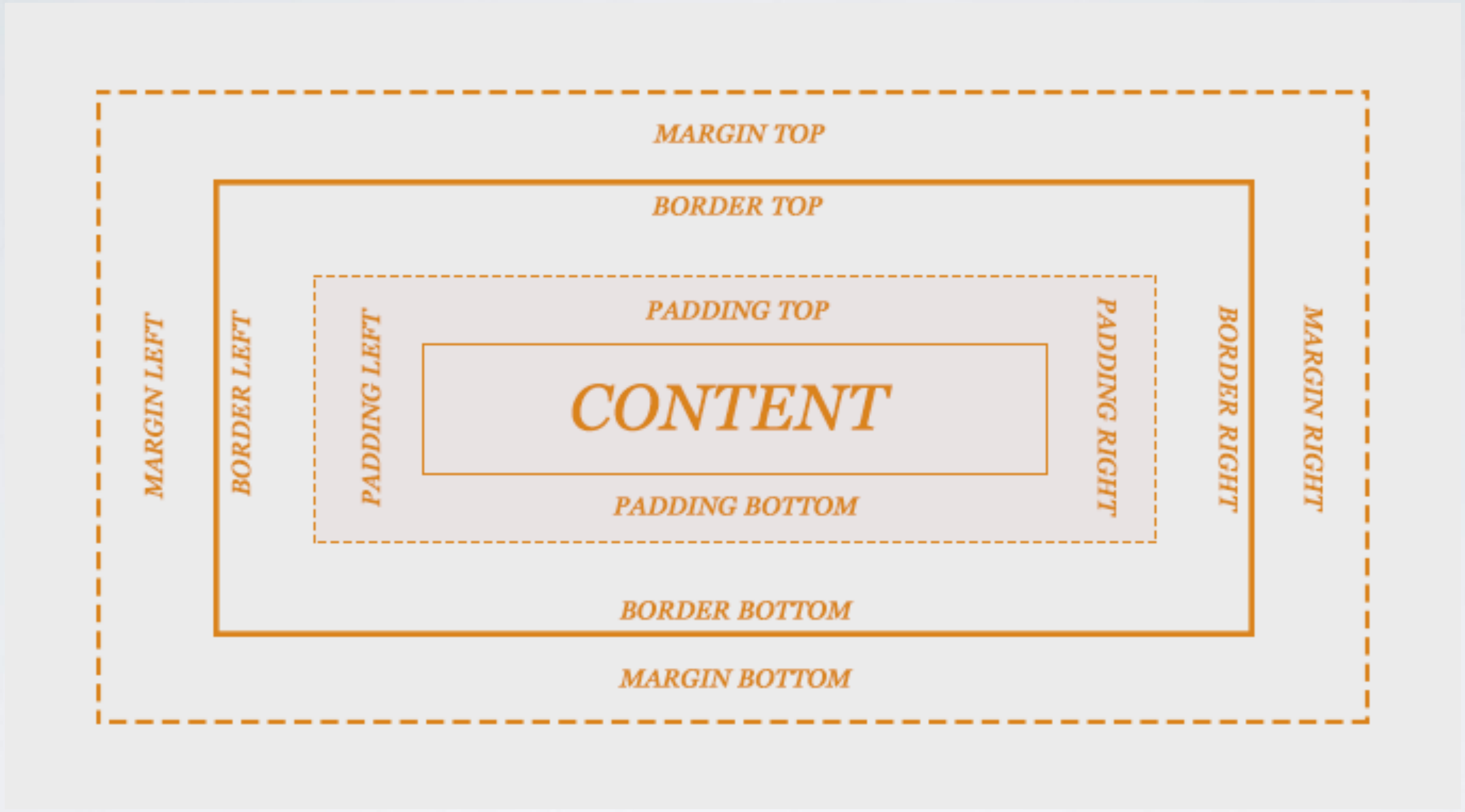
TEXT PROPERTIES

- text-align - justify blocks of text, e.g. {left, right, center, justify}
- text-decoration - {none, underline, overline, line-through, blink}
- text-transform - {non, capitalize, uppercase, lowercase}
- NB. Letter spacing, word spacing, line-height

LINKS

- Four states:
    - link - the normal state
    - visited - if the link has previously been followed
    - hover - while pointer is over link
    - active - whilst being clicked
- Define how links should look in above states, e.g. a:link {color:red}

BOX MODEL



- Each element of the web page is represented by a box
- margin - distance between edge of element and adjacent element
- padding - distance between edge of element and it's content
- border - frontier between padding and margin

- padding, border, & margin divided into four edges:
    - top, bottom, left, & right:
        - border-left, border-right, border-top and border-bottom
    - Same for margin and padding…
- Borders can be applied to all edges or to individual edges
- Border characteristics: {solid, dotted, dashed, double}
- Border width using a supplied value & colour

Edinburgh Napier UNIVERSITY

LAYOUTS

- Use HTML <span> and <div> block elements to assemble page layouts
    - span - inline, div - block
- Wrap around the different *blocks* of content for your page
- Give each a unique id attribute so that they can be identified, referenced, and subsequently positioned & styles by CSS
- NB. HTML requires IDs to be unique
- Gives us the basics for assembling layout patterns, e.g.
    - Two column layout: navbar + content
    - Three column layout: navbar + content + sidebar
    - NB. Consider header & footer as well

SUMMARY

- You should now:
  - Be able to explain why structure & presentation are treated separately
  - Understand how HTML & CSS are related
  - Have a basic grasp of the syntax & semantics of CSS

As with HTML (& later Javascript) we have only scratched the surface of what each technology can do. This is a foundation. Exploring what it can do ***for you*** *is your responsibility.*

Edinburgh Napier
UNIVERSITY

NEXT

- We'll revisit presentational aspects in a couple of weeks when we look at design & then again we look at accessibility
- But next is **Javascript**